Khushveen Kaur Umra

SE 317

22nd June 2022

Lab 5 Part I

According to the source, the user has the option to manually input the values to calculate the temperature and the relative humidity. It also allows the user to see the temperature and humidity trend based on the previous readings from the user. The code also allows the user to see what the status of humidity is based on the readings being reported.

Once you run the code, the user is asked to enter the values as <Temperature><Humidity>. They also have the option to reset the values if they wish to do so. The following snippets are the outputs one would see after running the code.

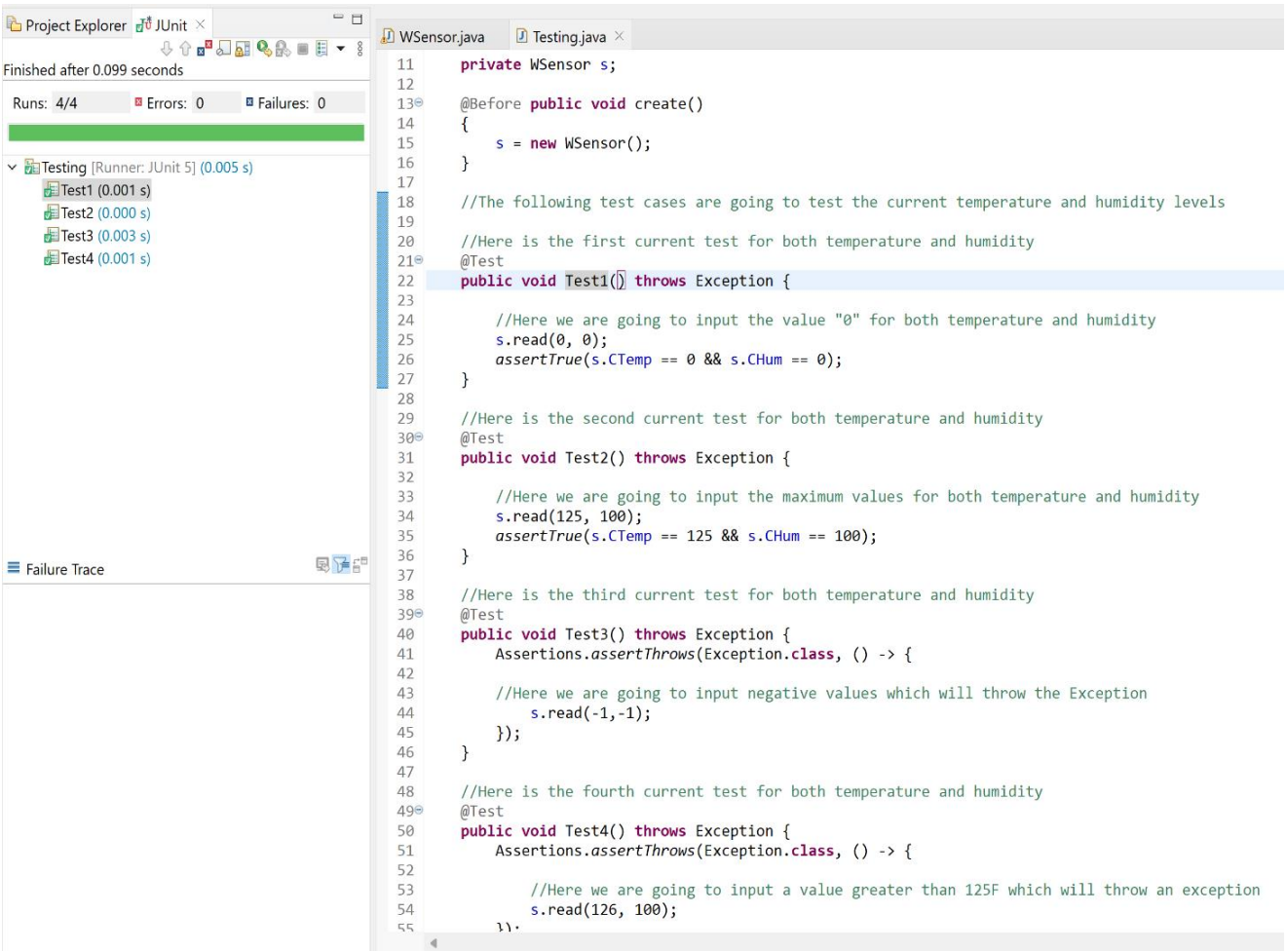
```
Console
WSensor (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (
Please wait for a few seconds after you enter the values
to allow the system to display the results!


Type 'reset' if you wish to reset the values!
Enter the values as <Temp><Humidity>:
Example : input 20 20


input 0 0
Relative Humidity:
Current Humidity: 0%
Maximum Humidity: 0%
Minimum Humidity: 0%
Humidity Trend: N/A

Temperature :
Current Temperature: 0F
Maximum Temperature: 0F
Minimum Temperature: 0F
Temperature Trend: N/A
Humidity Status: Low
input 20 20
Relative Humidity:
Current Humidity: 20%
Maximum Humidity: 20%
Minimum Humidity: 0%
Humidity Trend: Increasing

Temperature :
Current Temperature: 20F
Maximum Temperature: 20F
Minimum Temperature: 0F
Temperature Trend: Increasing
Humidity Status: Low
```

Here, you can see after you type “reset”, the trend and status values are now reset and no longer display the recorded values. You can then being to input more values, and the results will change.

```
Console
WSensor (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe
Maximum Temperature: 0F
Minimum Temperature: 0F
Temperature Trend: N/A
Humidity Status: Low
input 20 20
Relative Humidity:
Current Humidity: 20%
Maximum Humidity: 20%
Minimum Humidity: 0%
Humidity Trend: Increasing

Temperature :
Current Temperature: 20F
Maximum Temperature: 20F
Minimum Temperature: 0F
Temperature Trend: Increasing
Humidity Status: Low
reset
Relative Humidity:
Current Humidity: 20%
Maximum Humidity: 20%
Minimum Humidity: 20%
Humidity Trend: N/A

Temperature :
Current Temperature: 20F
Maximum Temperature: 20F
Minimum Temperature: 20F
Temperature Trend: N/A
Humidity Status: Low
input 70 70
Relative Humidity:
Current Humidity: 70%
Maximum Humidity: 70%
Minimum Humidity: 20%
Humidity Trend: Increasing

Temperature :
Current Temperature: 70F
Maximum Temperature: 70F
Minimum Temperature: 20F
Temperature Trend: Increasing
Humidity Status: Hi
```

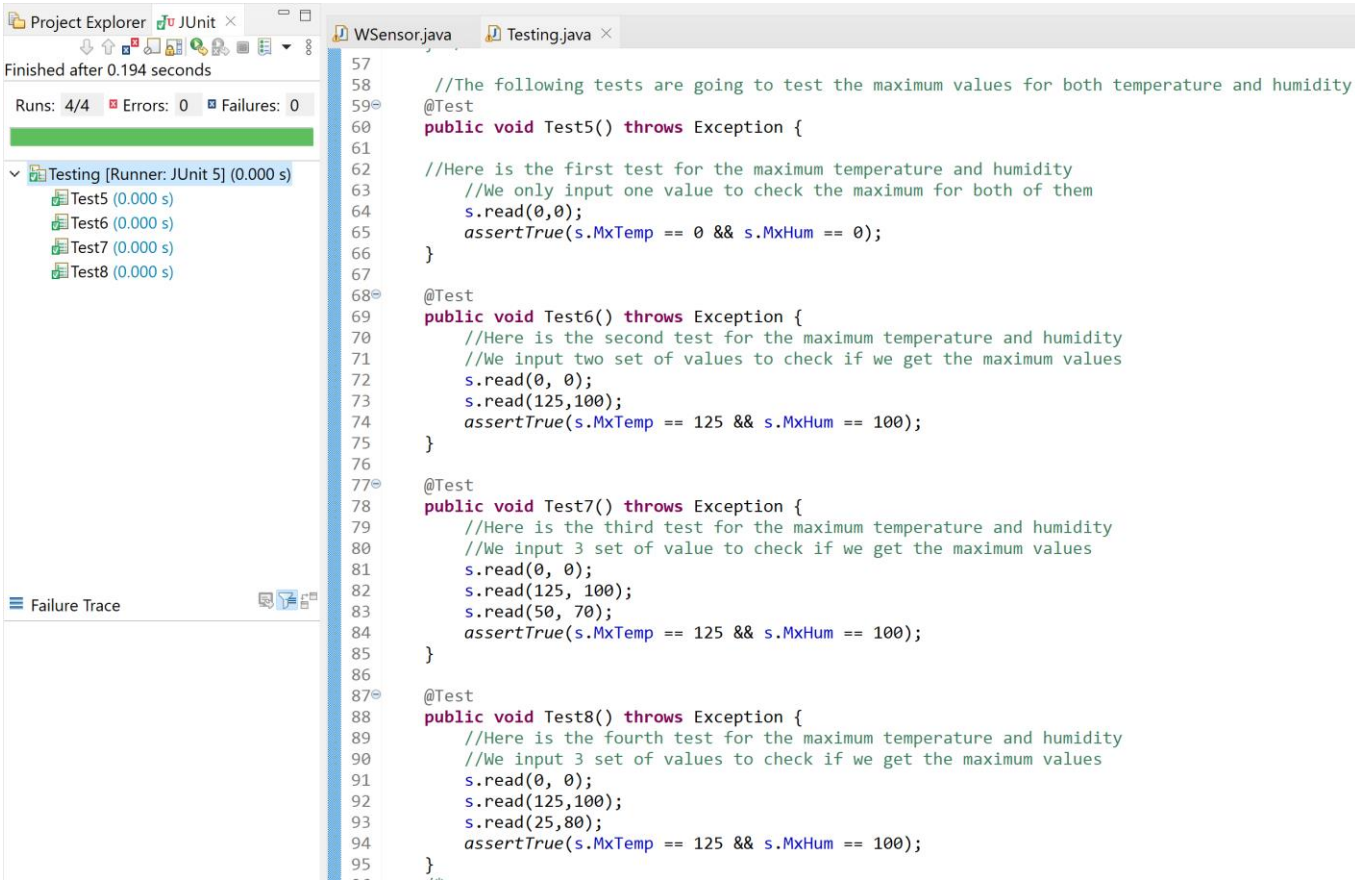
Lab 5 Part II

A) I) The following snippet are the test cases for the current temperature and current humidity. The first test case checks if the output (0,0) matches the current reading of (0,0). The second test case checks if the output (125, 100) matches the current reading. The rest two test cases throw exceptions due to values that are beyond the accepted range. Here the values represent (Temperature, Humidity)

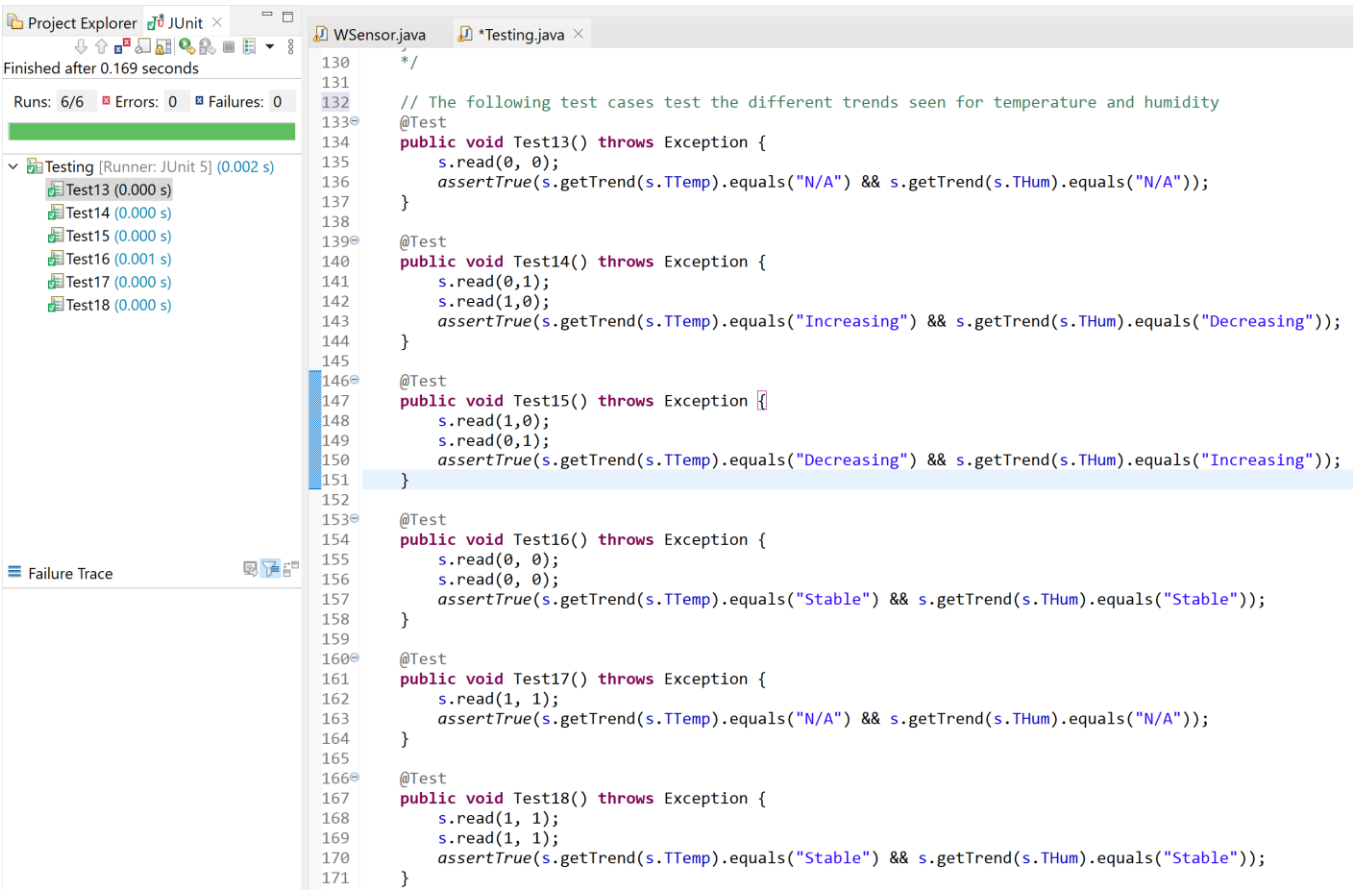



```java
    private WSensor s;

    @Before public void create()
    {
        s = new WSensor();
    }

    //The following test cases are going to test the current temperature and humidity levels

    //Here is the first current test for both temperature and humidity
    @Test
    public void Test1() throws Exception {

        //Here we are going to input the value "0" for both temperature and humidity
        s.read(0, 0);
        assertTrue(s.CTemp == 0 && s.CHum == 0);
    }

    //Here is the second current test for both temperature and humidity
    @Test
    public void Test2() throws Exception {

        //Here we are going to input the maximum values for both temperature and humidity
        s.read(125, 100);
        assertTrue(s.CTemp == 125 && s.CHum == 100);
    }

    //Here is the third current test for both temperature and humidity
    @Test
    public void Test3() throws Exception {
        Assertions.assertThrows(Exception.class, () -> {

        //Here we are going to input negative values which will throw the Exception
            s.read(-1,-1);
        });
    }

    //Here is the fourth current test for both temperature and humidity
    @Test
    public void Test4() throws Exception {
        Assertions.assertThrows(Exception.class, () -> {

            //Here we are going to input a value greater than 125F which will throw an exception
            s.read(126, 100);
        });
```

ii) The following test cases test the maximum temperature and humidity values. Test by test we increase the set of value in the test cases to see if we get the maximum values out of all the set of values. All the test cases run successfully with the correct maximum values.



iii) The following test cases test the different trends seen for both temperature and humidity. If the current value is greater than the previous value, it will indicate “Increasing”. If the current value is lower than the previous value, it will indicate “Decreasing”. If the previous value is the same value as the current value, it will indicate “Stable”. The other two test cases only have set of values for temperature and humidity, and since it as no value to compare, it will also indicate “Stable”.



iv) The following test cases test the humidity status.

Project Explorer | JUnit

Finished after 0.25 seconds

Runs: 3/3 Errors: 0 Failures: 0

Testing [Runner: JUnit 5] (0.001 s)
- Test19 (0.000 s)
- Test20 (0.001 s)
- Test21 (0.000 s)

WSensor.java | *Testing.java

```java
    @Test
    public void Test19() throws Exception {
        s.read(0,0);
        assertTrue(s.HumStatus(s.CHum).equals("OK"));
    }

    @Test
    public void Test20() throws Exception {
        s.read(0,1);
        s.read(0,1);
        assertTrue(s.HumStatus(s.CHum).equals("Hi"));
    }

    @Test
    public void Test21() throws Exception{
        s.read(0, 2);
        assertTrue(s.HumStatus(s.CHum).equals("Low"));
    }
}
```

B) Data-Driven Testing to avoid test code bloating

I) Style I – The following snippet uses the Style I testing, where we use only one test with the entire sequence for temperature and humidity values and see if we get the maximum and minimum values.

WSensor.java | Testing.java

```java
    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {66, 68, 69, 67, 63, 59, 53};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {53, 51, 48, 49, 54, 56, 56};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console

```
<terminated> WSensor (1) [Java Applicati
Relative Humidity:
Current Humidity: 56%
Maximum Humidity: 56%
Minimum Humidity: 48%
Humidity Trend: Stable

Temperature :
Current Temperature: 53F
Maximum Temperature: 69F
Minimum Temperature: 53F
Temperature Trend: Decreasing
Humidity Status: Hi
```

II) Style 2: The following snippets follow the Style 2 testing where the values were used as one pair. After running the code 7 times, you can see that maximum temperature and the maximum humidity recorded keeps changing according to the current values only.

WSensor.java × Testing.java

```java
    public int PHum;  // Previous Humidity
    public int MnTemp; //Minimum Temperature
    public int MnHum;  //Minimum Humidity
    public int MxTemp; //Maximum Temperature
    public int MxHum; //Maximum Humidity

    // Variables to display the trend seen in temperature and humidity changes

    public int TTemp; //Trend in temperature
    public int THum; //Trend in humidity

    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {66};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {53};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console ×

```
<terminated> WSensor (1) [Java App
Relative Humidity:
Current Humidity: 53%
Maximum Humidity: 53%
Minimum Humidity: 53%
Humidity Trend: N/A

Temperature :
Current Temperature: 66F
Maximum Temperature: 66F
Minimum Temperature: 66F
Temperature Trend: N/A
Humidity Status: OK
```

WSensor.java × Testing.java

```java
    public int PHum;  // Previous Humidity
    public int MnTemp; //Minimum Temperature
    public int MnHum;  //Minimum Humidity
    public int MxTemp; //Maximum Temperature
    public int MxHum; //Maximum Humidity

    // Variables to display the trend seen in temperature and humidity changes

    public int TTemp; //Trend in temperature
    public int THum; //Trend in humidity

    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {68};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {51};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console ×

```
<terminated> WSensor (1) [Java Ap
Relative Humidity:
Current Humidity: 51%
Maximum Humidity: 51%
Minimum Humidity: 51%
Humidity Trend: N/A

Temperature :
Current Temperature: 68F
Maximum Temperature: 68F
Minimum Temperature: 68F
Temperature Trend: N/A
Humidity Status: OK
```

WSensor.java | Testing.java

```java
    public int PHum;  // Previous Humidity
    public int MnTemp; //Minimum Temperature
    public int MnHum;  //Minimum Humidity
    public int MxTemp; //Maximum Temperature
    public int MxHum; //Maximum Humidity

    // Variables to display the trend seen in temperature and humidity changes

    public int TTemp; //Trend in temperature
    public int THum; //Trend in humidity

    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {69};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {48};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console

```
<terminated> WSensor (1) [Java A
Relative Humidity:
Current Humidity: 48%
Maximum Humidity: 48%
Minimum Humidity: 48%
Humidity Trend: N/A

Temperature :
Current Temperature: 69F
Maximum Temperature: 69F
Minimum Temperature: 69F
Temperature Trend: N/A
Humidity Status: OK
```

WSensor.java | Testing.java

```java
    public int PHum;  // Previous Humidity
    public int MnTemp; //Minimum Temperature
    public int MnHum;  //Minimum Humidity
    public int MxTemp; //Maximum Temperature
    public int MxHum; //Maximum Humidity

    // Variables to display the trend seen in temperature and humidity changes

    public int TTemp; //Trend in temperature
    public int THum; //Trend in humidity

    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {67};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {49};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console

```
<terminated> WSensor (1) [Java A
Relative Humidity:
Current Humidity: 49%
Maximum Humidity: 49%
Minimum Humidity: 49%
Humidity Trend: N/A

Temperature :
Current Temperature: 67F
Maximum Temperature: 67F
Minimum Temperature: 67F
Temperature Trend: N/A
Humidity Status: OK
```

WSensor.java | Testing.java

```java
    public int PHum;  // Previous Humidity
    public int MnTemp; //Minimum Temperature
    public int MnHum;  //Minimum Humidity
    public int MxTemp; //Maximum Temperature
    public int MxHum; //Maximum Humidity

    // Variables to display the trend seen in temperature and humidity changes

    public int TTemp; //Trend in temperature
    public int THum; //Trend in humidity

    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {63};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {54};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console

```
<terminated> WSensor (1) [Java A
Relative Humidity:
Current Humidity: 54%
Maximum Humidity: 54%
Minimum Humidity: 54%
Humidity Trend: N/A

Temperature :
Current Temperature: 63F
Maximum Temperature: 63F
Minimum Temperature: 63F
Temperature Trend: N/A
Humidity Status: OK
```

WSensor.java | Testing.java

```java
    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {59};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {56};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console

```
<terminated> WSensor (1) [Java A
Relative Humidity:
Current Humidity: 56%
Maximum Humidity: 56%
Minimum Humidity: 56%
Humidity Trend: N/A

Temperature :
Current Temperature: 59F
Maximum Temperature: 59F
Minimum Temperature: 59F
Temperature Trend: N/A
Humidity Status: Hi
```

WSensor.java | Testing.java

```java
    public static void main(String[] args) throws NumberFormatException, Exception {
        WSensor w = new WSensor();

        int [] temperature = {53};
        // {66, 68, 69, 67, 63, 59, 53};
        int [] humidity = {56};
        // {53, 51, 48, 49, 54, 56, 56};

        int length = temperature.length > humidity.length ? temperature.length:humidity.length;

                for(int i =0; i<length; i++)
                {
                    try {
                        w.read(temperature[i], humidity[i]); //To read the value the user input, as temperature and humidity
                    }

                    catch(Exception e)
                    {
                        throw new Exception("Invalid Input"); //To let the user know they have used the wrong format
                    }
                }

                    System.out.println(w.toString()); //To display the values input by the user
```

Console

```
<terminated> WSensor (1) [Java A
Relative Humidity:
Current Humidity: 56%
Maximum Humidity: 56%
Minimum Humidity: 56%
Humidity Trend: N/A

Temperature :
Current Temperature: 53F
Maximum Temperature: 53F
Minimum Temperature: 53F
Temperature Trend: N/A
Humidity Status: Hi
```

C) Refactoring

i) What is the difference between testing the 7 inputs in a sequence and testing them individually? How are the two test cases designed? (use narrative description, no test code needed.)

-> The difference between both the testing styles is the time it takes to do it. The second difference is that when we test them individually, it does not look at all the values and records the current values only. Hence, you do not get the accurate maximum and minimum values for the temperature and humidity values.

For testing 7 inputs in a sequence, you get the same result within a second without having to put in the values one by one. It is also more productive, as the test is there to find to the maximum and minimum values and indicates the highest and lowest value in the array.

By testing the inputs individually, it only looks at the value that is currently put in and gives the maximum and minimum according to the current value only. Hence you do not get the accurate information and is way less productive than the other style.

ii) I did not refactor my code, as I initially only used one algorithm to calculate the maximum, minimum values and the trend for the temperature and humidity, and hence all the calculations for being used for both the variables together.

iii) When I began to write the code, I was writing different functions for all the testing deliverables, and I realized that it was taking me a lot of time to understand everything and compiling them. Hence, I changed my code in such a way that all the calculations for the maximum, minimum values and the trend were the exact same for temperature and humidity. This also made my code smaller in comparison, and easier to read. For this I used the Math.min and Math.max functions to calculate the values and display them accordingly.

iv) I definitely feel that it would be easier to test a refactored code from another developer, as I would need to write way less test cases for all the functions, as everything is calculated in a single function. This would also save a lot of time, and I wouldn’t need to go through 1000 of lines of code just to realize that the calculations were similar for both temperature and humidity.

v) I would definitely prefer to test the refactored code by another developer, as I mentioned above that it would be much easier to test a code which is clean and well organized, as it would be much easier for me to go through the code and know what the functionality of the code is supposed to be.